Bilateral Space Video Segmentation

Nicolas Märki[1] Federico Perazzi[1,2] Oliver Wang[3] Alexander Sorkine-Hornung[2]
[1]ETH Zurich [2]Disney Research [3] Adobe Systems Inc.

Abstract

In this work, we propose a novel approach to video segmentation that operates in bilateral space. *We design a new energy on the vertices of a regularly sampled spatio-temporal bilateral grid, which can be solved efficiently using a standard graph cut label assignment. Using a bilateral formulation, the energy that we minimize implicitly approximates long-range, spatio-temporal connections between pixels while still containing only a small number of variables and only local graph edges. We compare to a number of recent methods, and show that our approach achieves state-of-the-art results on multiple benchmarks in a fraction of the runtime. Furthermore, our method scales linearly with image size, allowing for interactive feedback on real-world high resolution video.*

Figure 1. Example results of our bilateral space video segmentation which automatically propagates a user provided mask on the first frame (left column) through the complete video (remaining columns). Thanks to the efficiency of our method, errors in the later frames can be easily fixed in an interactive manner.

1. Introduction

Video segmentation is a fundamental problem in computer vision, and a key component of numerous applications, including localized video editing, color grading, and compositing. Furthermore, semantically meaningful clustering of video input is a crucial step in higher-level vision problems such as scene understanding, learning object class models, and video summarization [15, 16].

There are many similar and related definitions of what specifically "video segmentation" refers to, but for the purpose of this work, we consider the problem of finding a binary partitioning of pixels into foreground and background classes.

In general, this problem must be addressed semi-automatically, as the task itself requires a high level understanding of both the scene *and* the implicit goals of the user. This is because defining what constitutes a "foreground object" is often application specific, and the same video could have multiple valid solutions. For this reason, we consider a semi-supervised approach that uses sparse user-given cues to produce a video segmentation. These cues can be either a mask specified on one or a few key-frames, in which case the problem can be described as *mask propagation*, or a sparse set of labels specified by user clicks or strokes, which are then interpolated through the video.

A crucial aspect of semi-automatic video segmentation methods is *responsiveness*. A user expects instant feedback, and any computation delays present significant challenges to the adoption of these technologies. This is one of the key reasons that segmentation related tasks, such as rotoscoping, form the bulk of manual labor, and therefore associated costs, of video effects. In this work, we present a highly *efficient* method for user-guided video segmentation that is able provide iterative feedback in a fraction of the time of previous approaches, while still generating high quality results in semi-supervised applications, as demonstrated on multiple benchmarks.

We accomplish this by performing the segmentation in "bilateral space", which is a high dimensional feature space, originally proposed for accelerated bilateral filtering [8], and recently extended to computing depth from stereo triangulation [4]. We describe a novel energy on a "bilateral grid" [8], a regular lattice in bilateral space, and infer labels for these vertices by minimizing an energy using graph cuts. Processing on the bilateral grid has several advantages over other approaches. First, the regular and data-independent structure allows for a more efficient mapping from image to bilateral space (and vice versa) than super-pixels or k-means clustering approaches. Second, it allows for flexible

interpolation schemes that lead to soft assignments of pixels to multiple intermediate variables. And finally, a bilateral representation allows us to infer labels on a simple, locally connected graph, while still enforcing large spatio-temporal neighborhood regularization, which would be intractable to solve directly. We show that the combination of these advantages significantly improves segmentation quality, and importantly, allows us to segment video data, generating temporally consistent results with robustness to object and camera motion.

In summary, we present the first work to address video segmentation in bilateral space. Our approach contains several novel concepts, such as a fast "adjacent" interpolation scheme for high-dimensional grids, and a novel energy formulation that is justified by an analysis of locally connected labeling in bilateral space. Our method is highly efficient and scales linearly with image resolution, allowing us to process 1080p video with only minor increases in runtime. We compare our mask propagation to a number of existing approaches using multiple publicly available datasets, and demonstrate that using this simple to implement method we can achieve state-of-the-art results. While these comparisons are computed without user interaction, we note that the real strength of our approach is that it enables an interactive interface, due to the computational efficiency, which we show in a supplemental video.

2. Related Work

Graph Based Video Segmentation Images and videos naturally lend themselves to a regular graph structure where edges connect neighboring pixels in either a spatial or spatio-temporal configuration. Video segmentation can then be formulated as an optimization problem that tries to balance a coherent label assignment of neighboring vertices, while complying to a predetermined object model or user constraints. Graph-cuts techniques have long been used to efficiently solve this problem, both for image [6, 32] and video segmentation [11, 17, 21, 28, 30, 35].

Building on this general framework, subsequent methods have lowered the computational cost by reducing the number of nodes in the graph using clustering techniques such as a per-frame watershed algorithm [21, 28], mean-shift segmentation [35], or spatio-temporal superpixels [30]. However, these methods still do not achieve interactive rates due to costly clustering steps, and allow only rough user control [21], or require expensive per-pixel refinement on each frame [35]. Additionally, the above clustering methods can fail in regions with poorly defined image boundaries. Recently, efficient fully connected graphs have been exploited to improve robustness to long-range, possibly occluded connections [27].

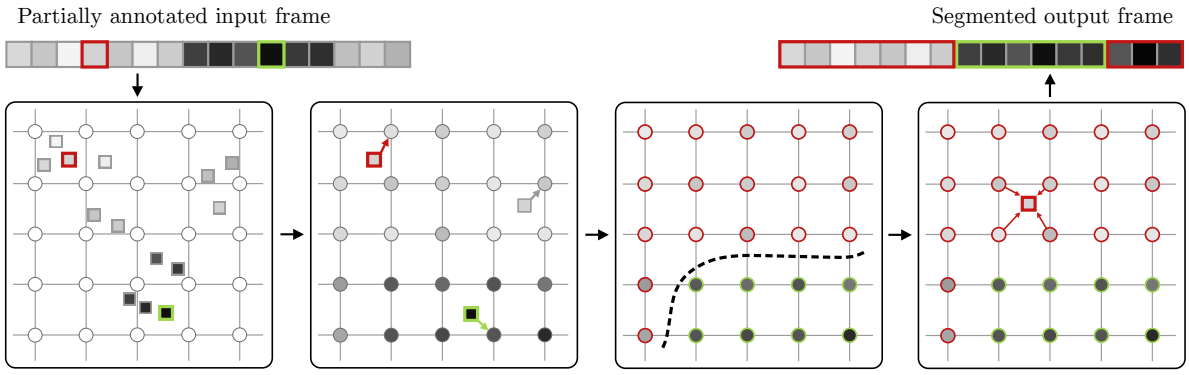
Other Semi-Supervised Approaches Besides graph-based methods, other approaches have proposed solutions to the video segmentation problem using, for example, optical flow or nearest neighbor fields to propagate silhouettes or masks over multiple key frames [2, 3, 10, 13, 18]. Video SnapCut [3] uses overlapping local classifiers that predict the foreground probability, which are propagated and refined over time. This approach was extended to a combination of local and global classifiers [38] to improve robustness. Dondera et al. [11] apply the spectral clustering method of [22] on a graph of super-pixels in a 3D video volume. An initial segmentation is obtained without additional input, the user can then add constraints to correct the solution. Labels are then inferred using a conditional random field formulation.

Recently, Fan et al. [13] proposed a method that propagates masks using nearest neighbor fields, and then refines the result with active contours on classified edge maps. As this is currently the top performing method in many cases, we use it as a basis for our comparisons.

These approaches solve the mask propagation locally, which makes enforcing global spatio-temporal constraints difficult. As opposed to this, our method has the benefits of a fully global solve, while operating on a reduced space, which yields a highly efficient solution.

Unsupervised Approaches By making certain assumptions about the application scenario, some methods have presented fully unsupervised techniques for video segmentation. These approaches use features such as clustering of point trajectories [7], motion characteristics [23], appearance [16, 19], or occlusion cues [33] to hypothesize foreground object locations. Faktor et al. [12] achieved state-of-the-art results by diffusing such hypotheses on a non-local graph. Similarly, Wang et al. [36] aggregate spatio-temporal saliency information [25] to infer the object appearance model. While automatic video segmentation methods certainly have advantages, they are only valid in restricted use cases where the desired "foreground" regions have notably different characteristics. For general purpose, high quality video segmentation, we instead focus on the user-assisted case, but note that our method could be combined with any existing automatic foreground model.

Bilateral Space Bilateral filtering has been widely used for edge-adhering image processing operations [24]. Chen et al. [8] introduced the *bilateral grid* as a data structure to speed up bilateral filtering. This approach lifts pixels into a higher-dimensional space based on position and color, after which bilateral filtering can be performed as a standard Gaussian filter. The advantage is that the resolution of the bilateral grid can be significantly lower than the number of input pixels, thereby providing an effective means of ac-



(a) Lifting (b) Splatting (c) Graph cut (d) Slicing

Figure 2. Our pipeline, demonstrated on a 1D example. Pixels are lifted into a 2D feature space (a), with two user assigned labels (red and green highlighted pixels). Values are accumulated on the vertices of a regular grid (b), a graph cut label assignment is computed on these vertices (c), and finally pixel values are sliced at at their original locations (c), showing the final segmentation (again, red and green boundaries).

celeration. This idea was later generalized to high dimensional simplexes [1], and has been used beyond filtering operations for edge preserving painting [8], and accelerating stereo matching [4,31]. Our method draws inspiration from these approaches. In particular, we extend the work of [4], and describe an energy that when solved using graph cuts, can be used to achieve a high quality video segmentation.

3. Method

Let $\mathcal{V} : \Omega \rightarrow \mathbb{R}^3$ be a color video, defined on a finite discrete domain $\Omega \subset \mathbb{R}^3$. Given some user input as a set of known foreground and background pixels, $FG, BG \subset \Omega$, we seek a binary mask $\mathcal{M} : \Omega \rightarrow \{0, 1\}$ that labels each pixel of the video either as background or foreground.

Our approach makes use of a bilateral grid [8], Γ, consisting of regularly sampled vertices $\mathbf{v} \in \Gamma$. The mask $\mathcal{M}$ is computed in four main stages, (summarized in Figure 2): by *lifting* pixels into a higher dimensional feature space (Section 3.1), *splatting* them onto regularly sampled vertices (Section 3.2), computing a *graph cut* label assignment (Section 3.3), and *slicing* vertex labels back into pixel space (Section 3.4).

3.1. Lifting

The first step is to embed each pixel $\mathbf{p} = [x, y, t]^\mathsf{T}$ in a higher d-dimensional feature space, for example by concatenating YUV pixel color and spatial and temporal coordinates:

$$\mathbf{b}(\mathbf{p}) = [c_y,\, c_u,\, c_v,\, x,\, y,\, t]^\mathsf{T} \in \mathbb{R}^6 \qquad (1)$$

In this bilateral space, Euclidean distance encodes both spatial proximity and appearance similarity. We evaluated a number of feature spaces, generalized as the concatenation of appearance features $\mathcal{A}(\mathbf{p})$ and position features $\mathcal{P}(\mathbf{p})$, and interestingly found that state-of-the-art results can be achieved by simply extending traditional 5D bilateral features with a temporal dimension, which is very efficient due to the low dimensionality.

3.2. Splatting

Instead of labeling each lifted pixel $\mathbf{b}(\mathbf{p})$ directly, we resample the bilateral space using a regular grid [4, 8] and compute labels on the vertices of this grid. The process of accumulating values on the bilateral space vertices is known as "splatting". For each vertex $\mathbf{v} \in \Gamma$, a weighted sum of lifted pixels $\mathbf{b}(\mathbf{p})$ is computed as:

$$S(\mathbf{v}) = \sum w(\mathbf{v}, \mathbf{b}(\mathbf{p})) \cdot (\hat{\mathbf{p}}) \qquad (2)$$

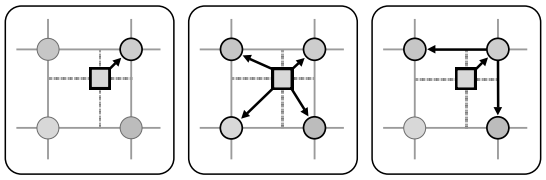
where

$$\hat{\mathbf{p}} = (\mathbb{1}_{FG}(\mathbf{p}), \mathbb{1}_{BG}(\mathbf{p}), 1) \qquad (3)$$

and $\mathbb{1}_{\times}(\mathbf{p})$ is an indicator function that is 1 iff $\mathbf{p} \in \times$.

The weight function $w(\mathbf{v}, \mathbf{b}(\mathbf{p}))$, determines the range and influence that each lifted pixel $\mathbf{b}(\mathbf{p})$ has on the vertices of Γ. Prior work has used a nearest neighbor (NN) indicator [4] or multi-linear interpolation weights [8]. Importantly, these approaches have limited support, (1 nonzero vertex for each pixel using NN, and 2^{d-1} for multi-linear), which is necessary for computation and memory efficiency. The NN approach is the fastest, but can lead to blocky artifacts, while the multi-linear interpolation is slower, but generates higher quality results. We propose an *adjacent* interpolation that provides a good compromise between the two, yielding high quality results, but with a linear growth in the number of non-zero weights as a function of feature space dimension, as opposed to the exponential growth of the multi-linear case (Figure 3).

The idea behind adjacent weighting is that with multi-linear interpolation, weights quickly decrease for vertices



(a) Nearest (b) Multi-linear (c) Adjacent

Figure 3. Different interpolation schemes. Adjacent interpolation scales significantly better to higher dimensionality when compared to multi-linear interpolation, with only a small reduction in quality.

that differ from the nearest neighbor $N_{\mathbf{b}(\mathbf{p})}$ in many dimensions. More precisely,

$$w_l(\mathbf{v}, \mathbf{b}(\mathbf{p})) \leq 0.5^{|\mathbf{v} - N_{\mathbf{b}(\mathbf{p})}|_0} \tag{4}$$

presents an upper bound for the weight, because each factor of the linear interpolation is smaller than 0.5 if for that dimension $\mathbf{v}_i$ is not the integer value that $\mathbf{b}_i(\mathbf{p})$ was rounded to. We use this bound to skip weight computation where the result would have been small anyway:

$$w_a(\mathbf{v}, \mathbf{b}(\mathbf{p})) = \begin{cases} \prod_{i=1}^{d} \left|\mathbf{v}_i - N_{\mathbf{b}(\mathbf{p})}\right| & \text{if } \mathbf{v} \in A_{\mathbf{b}(\mathbf{p})} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We found that interpolation between the nearest neighbor and vertices that differ in only one dimension (the set of adjacent vertices $A_{\mathbf{b}(\mathbf{p})}$) already produces significantly better results than hard nearest neighbor assignments with only a minor increase in runtime.

3.3. Graph Cut

We now seek binary labels $\boldsymbol{\alpha}$, that mark each vertex $\mathbf{v}$ as foreground, $\alpha_{\mathbf{v}} = 1$, or background, $\alpha_{\mathbf{v}} = 0$.

We compute these labels by constructing a graph $\mathcal{G} = (\mathit{\Gamma}, \mathcal{E})$ where the vertices are the vertices in the bilateral grid, and edges connect immediate neighbors (e.g., 4 neighbors when $d = 2$, 6 neighbors when $d = 3, \ldots$). We then define an energy based on the assumption that the label assignment is smooth in bilateral space:

$$E(\boldsymbol{\alpha}) = \sum_{\mathbf{v} \in \mathit{\Gamma}} \theta_{\mathbf{v}}(\mathbf{v}, \alpha_{\mathbf{v}}) + \lambda \sum_{(\mathbf{u},\mathbf{v}) \in \mathcal{E}} \theta_{\mathbf{uv}}(\mathbf{u}, \alpha_{\mathbf{u}}, \mathbf{v}, \alpha_{\mathbf{v}}) \tag{6}$$

$\theta_{\mathbf{v}}$ is the unary term, $\theta_{\mathbf{uv}}$ is the pairwise term, and λ is a weight that balances the two.

The unary term $\theta_{\mathbf{v}}$ models deviations from the supplied user input. As we invert the splatting step to retrieve final pixel labels, the splatted value $S_{BG}(\mathbf{v})$ expresses the total cost of assigning $\mathbf{v}$ to foreground, $\alpha_{\mathbf{v}} = 1$, and $S_{FG}(\mathbf{v})$ the cost of assigning it to background, $\alpha_{\mathbf{v}} = 0$, respectively.

$$\theta_{\mathbf{v}}(\mathbf{v}, \alpha_{\mathbf{v}}) = (1 - \alpha_{\mathbf{v}}) \cdot S_{FG}(\mathbf{v}) + \alpha_{\mathbf{v}} \cdot S_{BG}(\mathbf{v}) \tag{7}$$

The pairwise term $\theta_{\mathbf{uv}}$ attempts to ensure that neighboring vertices are assigned the same label. In order to derive $\theta_{\mathbf{uv}}$, we consider that the bilateral space graph $\mathcal{G}$ is equivalent to a densely connected pixel graph, where edge weight between pixels assigned to the same vertex is set to infinity (as it is impossible to assign them different labels in bilateral space). The edge weight between other pixels is then approximated by the distance of their respective vertices. With that in mind, it becomes clear that the weights between vertices need to be scaled by the total number of points $S_{\#}(\mathbf{u})$ and $S_{\#}(\mathbf{v})$ that have been assigned to the two vertices (we can retrieve $S_{\#}(\mathbf{u})$ and $S_{\#}(\mathbf{v})$ from the homogeneous (3rd) coordinate in Equation 4). That way, assigning different labels to two vertices is (approximately) equivalent to assigning the labels to all the original points and our pairwise term can be written as:

$$\theta_{\mathbf{uv}}(\mathbf{u}, \alpha_{\mathbf{u}}, \mathbf{v}, \alpha_{\mathbf{v}}) = g(\mathbf{u}, \mathbf{v}) \cdot S_{\#}(\mathbf{u}) \cdot S_{\#}(\mathbf{v}) \cdot [\alpha_{\mathbf{u}} \neq \alpha_{\mathbf{v}}] \tag{8}$$

where $g(\mathbf{u}, \mathbf{v})$ is a high-dimensional Gaussian kernel where the diagonal matrix $\mathit{\Sigma}$ scales each dimension to balance color, spatial and temporal dimensions:

$$g(\mathbf{u}, \mathbf{v}) = e^{-\frac{1}{2}(\mathbf{u}-\mathbf{v})^{\mathsf{T}} \mathit{\Sigma}^{-1} (\mathbf{u}-\mathbf{v})} \tag{9}$$

This formulation also reduces the complexity of the graph cut due to the fact that all vertices without any assigned pixels, $S_{\#}(\mathbf{v}) = 0$, are now completely excluded from any computation and thus need no representation in the graph. We can now efficiently apply a max-flow computation to find the vertex labeling with minimal energy [5].

Connectivity analysis So far we have assumed that increased connectivity leads to higher quality results. We validate this by conducting experiments where we compute a graph cut segmentation on a *per-pixel* (not bilateral) graph, as in [6]. We begin with just local neighbor edges (4 neighbors on a 2D graph), and increase the connectivity by connecting all points in an $n \times n$ window (Figure 4). This plot clearly shows that increasing connectivity leads to better results, but at an increased running time, as was also observed in [9, 12, 20].

3.4. Slicing

Given the foreground and background labels of the bilateral vertices, the final mask $\mathcal{M}$ is retrieved by slicing, i.e. interpolating grid labels at the positions of the lifted pixels in the output frame. We generally use the same interpolation scheme for both splatting and slicing, although a even more precise adjustment of the quality/speed trade-off is possible by choosing different interpolations.

$$\mathcal{M}(\mathbf{p}) = \sum_{\mathbf{v} \in \mathit{\Gamma}} w(\mathbf{v}, \mathbf{b}(\mathbf{p})) \cdot L(\mathbf{v}) \tag{10}$$

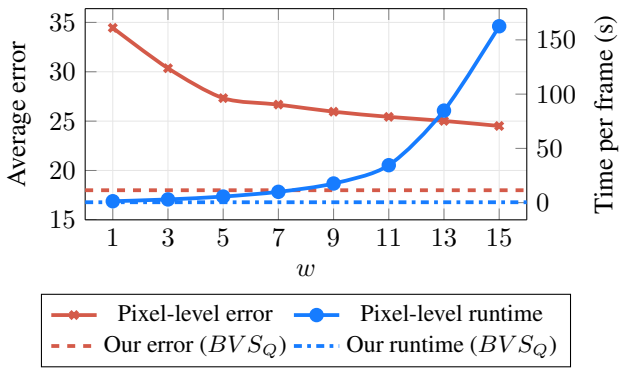


Figure 4. Mask propagation on a pixel-level graph with increasing neighborhood sizes w. Error decreases with larger neighborhoods at the expense of larger runtimes. Our approach (BVS_Q) is shown for comparison. We obtain lower error than even large window sizes, while being much faster as well.

Finally, we post-process each frame with a simple 3×3 median filter in order to remove minor high frequency artifacts that arise due to the solution being smooth in bilateral space, but not necessarily pixel space, however we note that a more sophisticated method like the geodesic active contours of [13] could also be applied.

4. Results and Evaluation

Implementation Our approach is implemented in Matlab, with C++ bindings for most time consuming routines. All our experiments were performed on a Mac Pro with a 3.5 GHz 6-Core Intel Xeon E5 CPU and 16 GB RAM. The measured timings include the complete pipeline except for IO-operations. Unlike many other approaches, we do not rely on pre-computed flows, edge maps or other information.

Parameters We evaluate two different sets of settings, one tuned for quality, BVS_Q, and the other for speed, BVS_S:

	BVS_Q (quality)	BVS_S (speed)
Feature space	YUV XY T	YUV XY T
Intensity grid size	35	15
Chroma grid size	30	10
Spatial grid size	$w/35, h/35$	$w/50, h/50$
Temporal grid size	2	2
Interpolation	Linear	Adjacent
Runtime	$0.37s$	$0.15s$

Our method can predict temporally global segmentations, and higher temporal resolutions allow for compensating for large degrees of object motion. However, this did not improve result quality on the benchmarks due to limited object motion, and the testing strategy of [13], where a single keyframe is propagated forward by multiple frames. In cases where user input is distributed temporally, e.g., in the interactive interface, we use a higher temporal grid size of $N = 5, \ldots, 15$.

We set the pairwise weight to $\lambda = 0.001$ for all results. The lifting stage also allows for different feature dimensions to be scaled independently of each other (Σ in Equation 9). For all results, we scale by 0.01, 0.5, 1.3, 1.5 the temporal (t), spatial (xy), the intensity (c_y) and the chroma ($c_u c_v$) dimensions respectively, but we didn't notice any particular dependency on the unary edge factor or the dimension scaling. All parameters could be tuned to achieve better results per benchmark or even per video, but we leave them fixed in all tests to represent a more real-world scenario.

Runtime Comparing runtime is difficult, with different code bases and levels of optimization, however, we give some average runtimes from our observations as a rough idea of the expected computational complexity. As many existing video segmentation methods take even up to one hour for a single frame, we compare only with the following fastest state-of-the-art methods: SEA: SeamSeg [29], JMP: JumpCut [13], NLC: Non-Local Consensus Voting [12], and HVS: Efficient Hierarchical Graph-Based Video Segmentation [14].

	BVS_Q	BVS_S	SEA	JMP	NLC	HVS
480p	$0.37s$	$0.15s$	$6s$	$12s$	$20s$	$5s$
1080p	$1.5s$	$0.8s$	$30s$	$49s$	$20s$	$24s$

Table 1. Approximate running time per frame for a number of fast methods with code available. Ours is roughly an order of magnitude faster than prior methods, and scales linearly with image size. NLC has mostly constant running time because it uses a fixed number of superpixels.

Our method computes 480p masks in as little as 0.15 seconds (Table 1) which is roughly an order of magnitude faster than all other approaches. Even if we trade speed for quality, our method still takes significantly less time than the second-fastest approach. Furthermore, the two most expensive steps, i.e. lifting and slicing, can be trivially parallelized since their output values only depends on color and position of individual pixels. Splatting can also be performed on concurrent threads, simply augmenting the grid with a small number of accumulators at bilateral vertices. The only stage that is not easily parallelizable is graph-cut, which anyway has small runtime due to the size and sparsity of the bilateral grid. Therefore we would expect a tuned GPU implementation to report substantial performance gains.

	BVS_Q	BVS_S	JMP	NLC	SEA	HVS
bear	**0.96**	0.93	0.93	0.91	0.91	0.94
blackswan	**0.94**	0.90	0.93	0.87	0.93	0.92
bmx-trees	**0.38**	0.29	0.23	0.21	0.11	0.18
bmx-bumps	0.43	0.41	0.34	**0.63**	0.20	0.43
breakdance-flare	0.73	0.59	0.43	**0.80**	0.13	0.50
breakdance	0.50	0.40	0.48	**0.67**	0.33	0.55
bus	**0.86**	0.84	0.67	0.63	0.75	0.81
dance-twirl	**0.49**	0.35	0.44	0.35	0.12	0.32
libby	**0.78**	0.61	0.29	0.64	0.23	0.55
dog	0.72	0.58	0.67	**0.81**	0.58	0.72
drift-chicane	0.03	0.01	0.24	0.32	0.12	**0.33**
drift-straight	0.40	0.21	**0.62**	0.47	0.51	0.30
mallard-water	**0.91**	0.82	0.75	0.76	0.87	0.70
mallard-fly	0.61	0.61	0.54	**0.62**	0.56	0.44
elephant	**0.85**	0.82	0.75	0.52	0.55	0.74
flamingo	**0.88**	0.72	0.53	0.54	0.58	0.81
goat	0.66	0.58	**0.73**	0.01	0.54	0.58
hike	0.76	0.82	0.66	**0.92**	0.78	0.88
paragliding	0.88	0.84	**0.95**	0.88	0.86	0.91
soccerball	**0.84**	0.57	0.10	0.83	0.65	0.07
surf	0.49	0.62	**0.94**	0.78	0.82	0.76
Average	**0.67**	0.56	0.61	0.64	0.56	0.60

Table 2. IoU score (higher is better) on a representative subset of the *DAVIS* benchmark [26], and the average computed over all 50 sequences.

	BVS_Q	BVS_S	RB	DA	SEA	JMP
animation	**0.78**	1.77	1.98	1.26	1.83	1.59
bball	**1.36**	3.29	1.55	1.71	1.90	1.61
bear	1.34	1.56	1.82	**1.07**	1.84	1.36
car	1.01	5.48	1.35	1.38	0.73	**0.54**
cheetah	**2.72**	3.56	7.17	3.99	5.07	4.41
couple	2.65	6.43	4.09	3.54	3.78	**2.27**
cup	**0.99**	4.54	3.72	1.34	1.19	1.16
dance	**5.19**	23.96	6.65	9.19	7.55	6.62
fish	**1.78**	4.06	2.80	1.97	2.54	1.80
giraffe	4.06	9.89	8.49	6.99	4.77	**3.83**
goat	2.68	4.87	3.68	2.57	3.30	**2.00**
hiphop	**3.21**	8.08	8.02	4.62	6.94	3.37
horse	3.60	16.32	3.99	4.14	3.00	**2.62**
kongfu	**1.97**	2.51	5.42	3.71	5.78	3.28
park	**2.35**	5.89	3.95	3.49	3.33	2.93
pig	2.15	3.18	3.86	**2.08**	3.39	2.97
pot	**0.62**	1.25	0.94	1.49	0.80	0.70
skater	**4.72**	11.23	6.33	5.33	5.09	4.89
station	2.07	8.55	2.53	2.01	2.37	**1.53**
supertramp	9.68	9.76	14.70	8.99	17.40	**6.17**
toy	0.66	7.16	1.02	1.32	0.70	**0.58**
tricking	**4.23**	5.57	42.20	9.71	11.90	5.02
Average	**2.72**	6.77	6.19	3.72	4.33	2.78

Table 3. Errors (lower is better) on the JumpCut benchmark for two transfer distances and several different methods as reported by [13].

4.1. Quantitative Evaluation

In order to evaluate our approach with respect to existing methods, we focus on the task of *mask propagation*, which has been widely used by previous work. Given a manual segmentation of the first frame, each method predicts subsequent frames, without any additional user input. Using this approach, we measured the performance on three different benchmark datasets.

DAVIS The dataset of Perazzi et al. [26] comprises a total of 50 high-resolution sequences alternating a wide range of object segmentation challenges such as occlusions, fast-motion and appearance changes. The dataset comes with per-frame, per-pixel manual annotations. Table 2 summarizes the results for a representative subset of *DAVIS* sequences and the average performance over the *entire* dataset. The full, per-sequence, evaluation can be found in the benchmark. While our approach scales linearly with image resolution, not all algorithms that we compare to are able to handle the full 1080p resolution, so we run comparisons on downscaled 480p versions of these sequences. We report the widely used intersection-over-union (IoU) metric, averaged over all frames in each sequence. As may be seen in Table 2, our method outperforms all other methods, achieving the best score on most of the videos and the best average score overall. Even with the faster, but less accurate configuration, our approach still performs comparably or better than several concurrent approaches [26].

JumpCut The recent method of Fan et al. [13] includes a dataset consisting of 22 videos with medium resolution and good per-frame ground truth masks. In addition to the methods mentioned above, we compare to RB: RotoBrush, based on SnapCut [13], and DA: Discontinuity-aware video object cutout [38]. As we do not have access to implementations for all methods reported on this dataset, we instead adapt our method to conform to the same testing strategy and error metric used in [13]. That is, propagating masks from multiple keyframes $0, 16, \ldots, 96$, over different transfer distances $(1, 4, 8, 16$ frames), and reporting error as follows:

$$Err = \frac{100}{n} \sum_{i=1}^{n} \frac{\#\text{ error pixels in i-th frame}}{\#\text{ foreground pixels in i-th frame}} \quad (11)$$

Overall, our method performs best on this benchmark, closely followed by JumpCut (Table 3).

We note that our approach uses a simple refinement step (3x3 median filter). However, we conducted an experiment using an active contour refinement, similar to JumpCut, and our result improved to 2.45 on average, with a running time

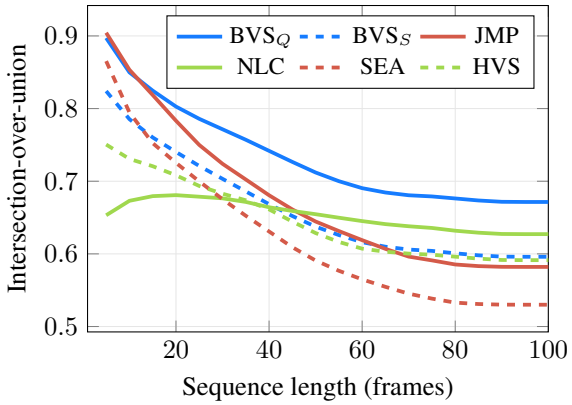


Figure 5. This plot shows how IoU (higher is better) decreases when a *single* mask is propagated over increasing numbers of frames. Our method degrades favorably when compared to other approaches. The NLC approach stays constant as it is an automatic method that doesn't depend on the input of the first frame.

of only $1s$ per frame. We additionally observe that many methods degrade in quality over long sequences, as errors accumulate over time. In contrast, our method scores better on long videos, experiencing less drift of the object region than other approaches (Figure 5).

SegTrack For the sake of completeness we also present an evaluation on the popular benchmark of Tsai et al. [34]. We additionally compare to: FST: Fast Object Segmentation in Unconstrained Video [23], DAG: Video object segmentation through spatially accurate and temporally dense extraction of primary object regions [37], TMF: Video segmentation by tracking many figure-ground segments [20], and KEY: Key-segments for video object segmentation [19]. In this case, it can be seen that our method clearly struggles to compete with existing approaches. This is most likely due to a combination of factors related to the low quality and resolution of the input videos, which lead to many mixed pixels that confuse the bilateral model. We also note that many of these methods were optimized with this dataset in mind, using different parameter settings per *sequence*. Instead, we use the same parameter settings for all three datasets. We also believe that the more recent datasets from JumpCut and our additional videos provide a more contemporary representation of video segmentation tasks.

4.2. Interactive Segmentation

It is important to note that while our method scores well on these two higher-resolution benchmarks, the real advantage is the fast running time, when used in an interactive framework. To demonstrate this, we built a simple prototype editor (Figure 7) in Matlab that allows a user to draw strokes on an image to mark foreground or background regions. After every stroke, the newly marked pixels are splatted to the bilateral grid and a global spatio-temporally solution is computed. Finally, the mask is sliced from the current frame and its outline is overlaid on the image. Please see the supplemental video for an example of this interaction.

	BVS$_Q$	BVS$_S$	NLC	FST	DAG	TMF	KEY	HVS
birdfall	0.66	0.40	**0.74**	0.59	0.71	0.62	0.49	0.57
cheetah	0.10	0.14	**0.69**	0.28	0.40	0.37	0.44	0.19
girl	0.89	0.87	**0.91**	0.73	0.82	0.89	0.88	0.32
monkeydog	0.41	0.38	0.78	**0.79**	0.75	0.71	0.74	0.68
parachute	0.94	0.92	0.94	0.91	0.94	0.93	**0.96**	0.69
Average	0.60	0.54	**0.81**	0.66	0.72	0.70	0.70	0.49

Table 4. Comparison of our method on the SegTrack dataset [34], using the IoU metric (higher is better).

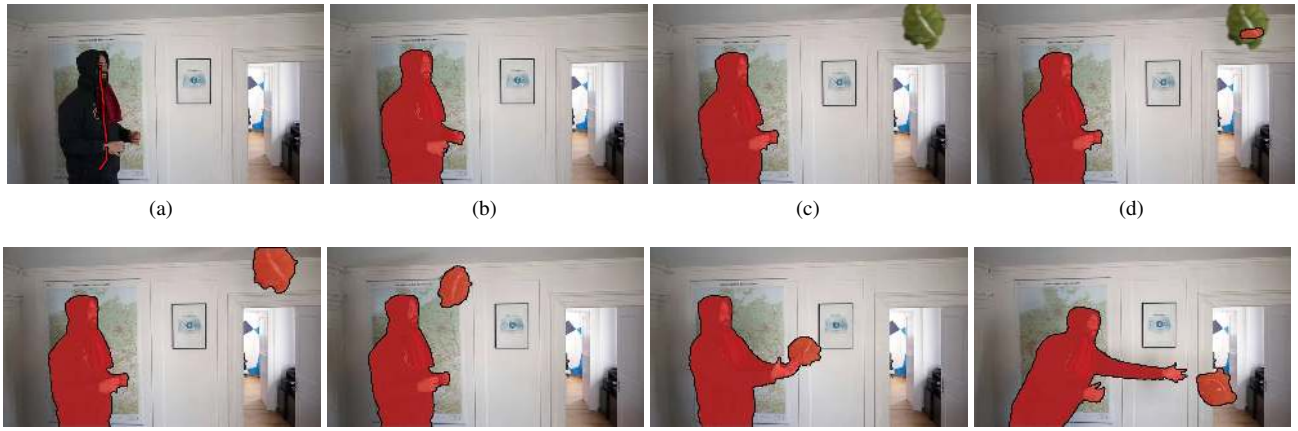
5. Discussion

In summary, we have shown how simple and well-understood video segmentation techniques leveraging graph cuts can yield state-of-the-art results when performed in bilateral space.

There are many exciting avenues for extending the research in this area. For example, one could consider alternate, more descriptive feature spaces in the lifting step. We made some initial experiments with using patches, and obtained marginally better results, but at the expense of higher running time. Additionally, while the bilateral representation can handle some degree of motion, it does not explicitly account for camera or object motion. One possibility is to warp pixels using their optical flow before splatting. Our initial experiments indicated that due to the instability of flow, such methods were unreliable; sometimes leading to large improvements in quality, but in other times made the results worse. These methods also rely on precomputing optical flow, which is costly. Nonetheless, explicitly exploring scene motion is a promising venue to future work.

Despite this, we believe that the method as presented here has many attractive qualities. It is simple to implement, parallelizable, and fast, all without sacrificing quality. This efficiency gain is not only vital to providing faster feedback to users, but is also important for extending to low computational power (mobile) devices, or large scale (cloud-based) problems, which will hopefully enable new applications.

Figure 6. Qualitative video segmentation results from three sequences of *DAVIS* [26] (*horsejump*, *stroller* and *soapbox*). The segmentation is computed non-interactively, given the first frame as initialization. Our method demonstrates robustness to challenging scenarios such as complex objects, fast-motion, and occlusions.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 7. Our interactive segmentation editor. Very simple input (a) is sufficient to infer an accurate foreground mask (b) and track it over time. As a new object enters the scene (c), the user can choose to add it to the foreground with an additional input stroke (d). The mask is then automatically propagated to to the other frames (e-h) without further corrections.

References

[1] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian KD-trees for fast high-dimensional filtering. *ACM Transactions on Graphics (TOG)*, 28(3):21, 2009. 3

[2] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *SIGGRAPH*, 2004. 2

[3] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video SnapCut: robust video object cutout using localized classifiers. *ACM Transactions on Graphics (TOG)*, 28(3):1, 2009. 2

[4] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In *CVPR*, 2015. 1, 3

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV*, 1999. 4

[6] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *ICCV*, 2001. 2, 4

[7] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010. 2

[8] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *SIGGRAPH*, 2007. 1, 2, 3

[9] I. Choi, M. Lee, and Y.-W. Tai. Video Matting Using Multiframe Nonlocal Matting Laplacian. In *ECCV*, 2012. 4

[10] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. In *SIGGRAPH*, 2002. 2

[11] R. Dondera, V. Morariu, Y. Wang, and L. Davis. Interactive video segmentation using occlusion boundaries and temporally coherent superpixels. In *2014 IEEE Winter Conference*

on Applications of Computer Vision (WACV), pages 784–791. IEEE, 2014. 2

[12] A. Faktor and M. Irani. Video Segmentation by Non-Local Consensus voting. *BMVC 2014*, pages 21.1–21.12, 2014. 2, 4, 5

[13] Q. Fan, F. Zhong, D. Lischinski, D. Cohen-Or, and B. Chen. JumpCut: Non-Successive Mask Transfer and Interpolation for Video Cutout. *SIGGRAPH Asia*, 2015. 2, 5, 6

[14] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 5

[15] A. Jain, S. Chatterjee, and R. Vidal. Coarse-to-Fine Semantic Video Segmentation Using Supervoxel Trees. *ICCV*, 2013. 1

[16] A. Khoreva, F. Galasso, M. Hein, and B. Schiele. Classifier Based Graph Construction for Video Segmentation. In *CVPR*, 2015. 1, 2

[17] P. Kohli and P. H. S. Torr. Dynamic Graph Cuts for Efficient Inference in Markov Random Fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(12):2079–2088, 2007. 2

[18] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. H. Gross. Practical temporal consistency for image-based graphics applications. *ACM Transactions on Graphics (TOG)*, 31(4):34, 2012. 2

[19] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011. 2, 7

[20] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013. 4, 7

[21] Y. Li, J. Sun, H.-Y. Shum, Y. Li, and J. Sun. Video object cut and paste. *ACM Transactions on Graphics (TOG)*, 24(3):595–600, July 2005. 2

[22] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002. 2

[23] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013. 2, 7

[24] S. Paris and F. Durand. A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach. In *ECCV*, 2006. 2

[25] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, 2012. 2

[26] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 6, 8

[27] F. Perazzi, O. Wang, M. H. Gross, and A. Sorkine-Hornung. Fully connected object proposals for video segmentation. In *ICCV*, 2015. 2

[28] B. L. Price, B. S. Morse, and S. Cohen. LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 2009. 2

[29] S. A. Ramakanth and R. V. Babu. Seamseg: Video object segmentation using patch seams. In *CVPR*, 2014. 5

[30] M. Reso, B. Scheuermann, J. Jachalsky, B. Rosenhahn, and J. Ostermann. Interactive Segmentation of High-Resolution Video Content Using Temporally Coherent Superpixels and Graph Cut. In *Advances in Visual Computing*. Springer International Publishing, Cham, Dec. 2014. 2

[31] C. Richardt, D. Orr, I. P. Davies, A. Criminisi, and N. A. Dodgson. Real-Time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid. In *ECCV*, 2010. 3

[32] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004. 2

[33] B. Taylor, V. Karasev, and S. Soatto. Causal Video Object Segmentation From Persistence of Occlusions. In *CVPR*, 2015. 2

[34] D. Tsai, M. Flagg, and J. M. Rehg. Motion Coherent Tracking with Multi-label MRF optimization. *BMVC*, pages 1–11, 2010. 7

[35] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Transactions on Graphics (TOG)*, 24(3):585–594, July 2005. 2

[36] W. Wang, J. Shen, and F. Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, 2015. 2

[37] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013. 7

[38] F. Zhong, X. Qin, Q. Peng, and X. Meng. Discontinuity-aware video object cutout. *ACM Transactions on Graphics (TOG)*, 31(6):175, Nov. 2012. 2, 6